Plano para Aplicativo de Microsserviços Java: aplicação CADU 2.0

VISÃO GERAL

Estrutura da Aplicação: Uma aplicação Java baseada em microsserviços e arquitetura hexagonal.

Microsserviços: Três serviços principais - “produtos”, “empresas” e “permissões” (na primeira PI).

Front-end: Um Single Page Application (SPA) baseado em Angular e arquitetura de microfrontends, hospedado no Azure Static Web App.

DETALHES DOS MICROSSERVIÇOS

Operações Básicas: Cada serviço implementará operações CRUD (Criar, Ler, Atualizar, Excluir).

Serviço de Produtos: Gerenciar dados e operações relacionados a produtos da B3.

Serviço de Empresas: Lidar com informações relacionadas a empresas (clientes B3), incluindo upload de arquivos para o Azure Storage Blob.

Serviço de Permissões: Controla as atividades permissões e controles de acesso de usuários (usuário comum, usuário máster e usuário interno [colaborador B3]).

TECNOLOGIAS APLICADAS

Back-end: Desenvolvido com Spring Boot para a criação de serviços RESTful.

Front-end: Single Page Application (SPA) criado em Angular e arquitetura de microfrontends, hospedado no Azure Static Web App.

Contêinerização: Uso de Docker para contêinerizar os serviços e uso do AKS (Azure Kunernetes Services) e ACR (Azure Container Registry) para deploy na nuvem.

Banco de Dados: Azure SQL Server fornecido pelo Azure Cloud Service para armazenamento de dados relacionais.

IMPLANTAÇÃO E ORQUESTRAÇÃO

Implantação: Os serviços serão implantados no Azure Kubernetes Service (AKS) para aproveitar seus recursos de escalabilidade e gerenciamento.

Orquestração de Containers: Gerenciada através do Kubernetes, garantindo uma implantação, dimensionamento e operação eficientes de containers de aplicativos em clusters de hosts.

COMUNICAÇÃO ENTRE SERVIÇOS

Azure Queue Storage: Utilizado para comunicação assíncrona entre serviços, aprimorando o desacoplamento e a escalabilidade.

FRONT-END

Angular SPA: Uma aplicação front-end robusta para interagir com os serviços back-end, com foco na experiência do usuário e no desempenho.

MELHORIAS ADICIONAIS DE ARQUITETURA

API Gateway: Implementação de um API Gateway para rotear solicitações, gerenciar balanceamento de carga e fornecer um único ponto de entrada para o ecossistema de microsserviços.

Descoberta de Serviços: Incorporação de ferramentas como Kubernetes DNS para descoberta dinâmica de serviços na arquitetura de microsserviços.

CONFIGURAÇÃO E GERENCIAMENTO

Spring Cloud Config: Para gerenciamento de configuração externalizada em um sistema distribuído, facilitando a consistência e a facilidade de mudanças em diferentes ambientes.

MONITORAMENTO, LOG E SEGURANÇA

Monitoramento: Integração de ferramentas como Prometheus para monitoramento em tempo real dos microsserviços.

Log: Implementação do ELK Stack (Elasticsearch, Logstash, Kibana) para registro e análise de logs eficientes.

Segurança: Ênfase no aprimoramento da segurança de APIs, implementação de mecanismos robustos de autenticação e autorização (como OAuth2 e JWT) e proteção de recursos do Azure.

RESILIÊNCIA E TOLERÂNCIA A FALHAS

Padrão de Disjuntor: Uso do Spring Cloud Hystrix para tornar os serviços resilientes a falhas e prevenir efeitos de falha em cascata.

OTIMIZAÇÃO DE PERFORMANCE

Cache de Dados: Implementação de estratégias de cache para dados acessados com frequência para melhorar o desempenho e reduzir a latência.

CONCLUSÃO

Este plano abrangente descreve uma aplicação de microsserviços Java robusta, escalável e segura, aproveitando tecnologias modernas e melhores práticas. A arquitetura foi projetada para ser resiliente, facilmente manutenível e capaz de lidar com o crescimento e as mudanças de requisitos.